Stats 4CI3 - Assignment 4

Tommy Flynn (400121496)

March 7, 2021

Question 1:

Using the following code

```
set.seed(1234)
n = 1000
U1 = runif(n,0,1)
U2 = runif(n,0,1)
U3 = runif(n,0,1)
mean(1/(1+U1))
mean((U1+U2+U3)^0.5)
```

we have

```
[1] 0.6900332
[1] 1.208232
```

Therefore,

a) $E[1/(1+U_1)] = 0.6900332$

b) $E[(U_1+U_2+U_3)^{\frac{1}{2}}] = 1.208232$

Question 2:

Here is the code I used for this question.

```
set.seed(1234)
p = 0.01
m = 1000
n = 1000
X = rbinom(n,m,p)
Z = (X-m*p)/(sqrt(m*p*(1-p)))
qqnorm(Z)
abline(0,1,col="red")
```

a) Here is the QQ plot of Z with the line $y = x$ in red. The plot follows the line very loosely so the normality assumption is not strong here.

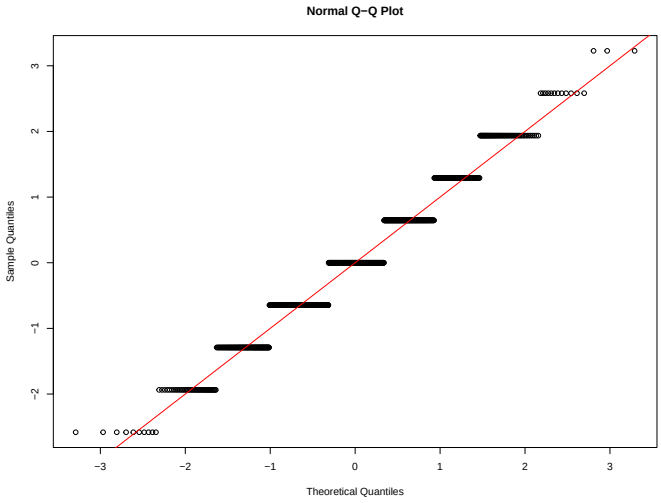

Figure 1: $m = 10, p = 0.4, n = 1000$

b) Repeating for $m \in \{10^2, 10^3, 10^4\}$ we have the following.
From the figures below we see that as m increase the stronger the normality of Z, it seems fairly reasonable at $m = 1000$.

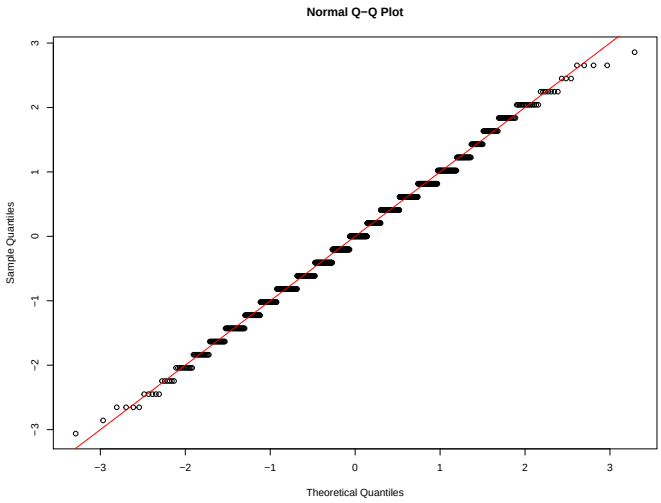

Figure 2: $m = 100, p = 0.4, n = 1000$

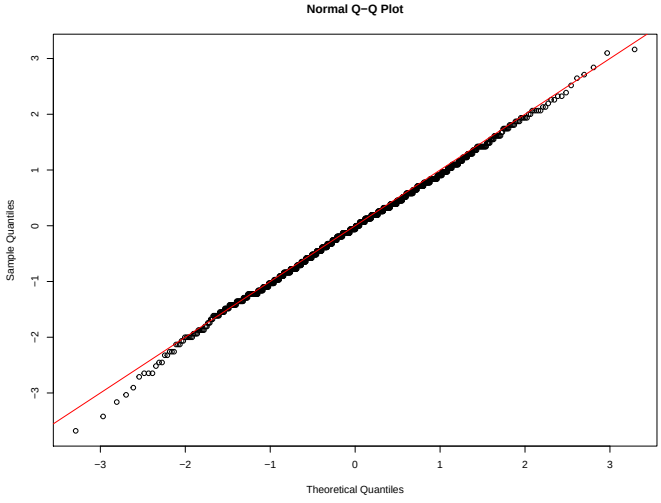

Figure 3: $m = 1000, p = 0.4, n = 1000$

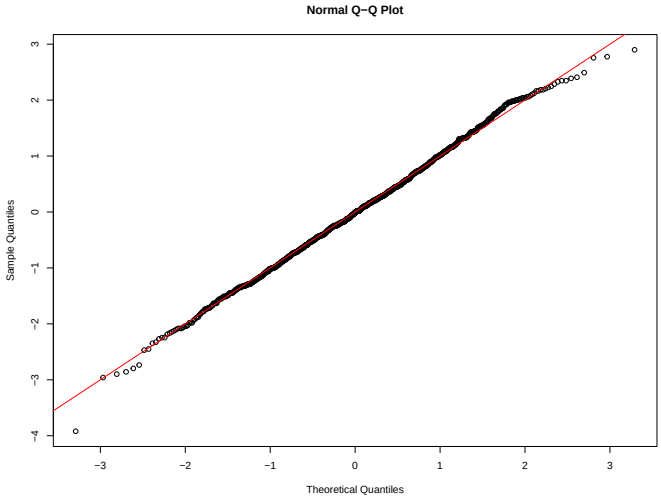

Figure 4: $m = 10000, p = 0.4, n = 1000$

c) Lastly, we repeat with $m = 1000, p = 0.01$.

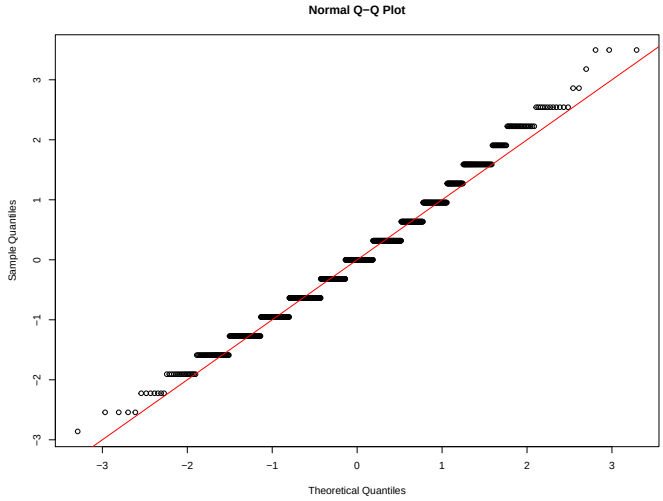

Figure 5: $m = 1000, p = 0.01, n = 1000$

This plot appears to mimic the plot where $m = 100, p = 0.4$. Therefore, the larger m is and the closer p is to 0.5 the more normal Z will be.

Question 3:

Below is the relevant code and output for each part.

a)

```
set.seed(1234)
P = t(matrix(c(0.50, 0.50, 0.00, 0.00, 0.00, 0.00,
               0.25, 0.50, 0.25, 0.00, 0.00, 0.00,
               0.00, 0.25, 0.50, 0.25, 0.00, 0.00,
               0.00, 0.00, 0.25, 0.50, 0.25, 0.00,
               0.00, 0.00, 0.00, 0.25, 0.50, 0.25,
               0.00, 0.00, 0.00, 0.00, 0.50, 0.50), nrow=6,ncol=6))
```

b)

```
s = array(dim = 50000)
```

c)

```
s[1] = 3
for (j in 2:50000)
  {
    new = sample(1:6,1,prob = P[s[j-1],])
    s[j] = new
}
```

d)

```
table(s[1:500])/500
table(s[1:2000])/2000
table(s[1:8000])/8000
table(s[1:50000])/50000
```

```
    1     2     3     4     5     6
0.078 0.198 0.184 0.228 0.208 0.104
```

```
     1      2      3      4      5      6
0.0915 0.1805 0.1620 0.2020 0.2390 0.1250
```

```
       1        2        3        4        5        6
0.089500 0.175750 0.174625 0.208625 0.233875 0.117625
```

```
      1       2       3       4       5       6
0.09398 0.18768 0.19312 0.20560 0.21188 0.10774
```

e) We can see from the tables above that as the step count increase we are approaching the stationary distribution given by the following.

```
round(table(s[1:50000])/50000,1)
```

```
  1   2   3   4   5   6
0.1 0.2 0.2 0.2 0.2 0.1
```

Question 4:

a)

1. Initialize $X^{(0)} = x_0$ and candidate density $q(y|x)$
for $t = 0 \dots n$
2. Given $X^{(t)} = x$ simulate $Y_t = y$ from $q(y|x)$
3. Compute $\rho(x, y) = \min\{\frac{f(y)}{f(x)} \frac{q(x|y)}{q(y|x)}, 1\}$
4. If $\rho(x, y) < 1$ then generate $U_t \sim \text{Uniform}(0, 1)$
5. Set $X^{(t+1)}$
$= y$ if $\rho(x, y) = 1$ or $U_t < \rho(x, y)$
$= x$ if $U_t \geq \rho(x, Y_t)$

b) The difference between the Metropolis-Hastings algorithm and the Independent Metropolis-Hastings algorithm is the former is dependent on the current state while the latter is not. In particular, this means that the candidate density is of the form $q(y|x) = g(y)$ for the independent algorithm.

c) For the independent Metropolis-Hastings algorithm to work well, we need the candidate density to have the same support as f. We would also like $\frac{f(x)}{g(x)}$ to be bounded and for g to be as similar to f as possible while being easy to sample from.

d) The difference between the outputs of the Metropolis-Hastings algorithm and the Accept-Reject algorithm is the former is a dependent sequence and the latter is an independent and identically distributed sequence.

e) The main consequence of choosing a bad candidate density function is the chain can halt at a point x where $f(x) >> g(x)$ for long periods of time.

f) Compared to the chain from an Independence Metropolis-Hastings algorithm, the Random Walk Metropolis-Hastings algorithm makes much smaller and more rapid movements.

g) The importance of the scale for a Random Walk Metropolis-Hastings algorithm is its significant influence on convergence. If the scale is too small then convergence will be quite slow and if the scale is too large the algorithm can mimic the Independent Metropolis-Hastings algorithm with the issue of halting. Thus, one must carefully choose the scale parameter for an effective run.

h) The main difference between a Markov Chain produced with the Gibbs Sampling technique, as compared to the Metropolis-Hastings algorithm is the Gibbs Sampler is guaranteed to move at every iteration while the Metropolis-Hastings algorithm is not.

i) The easiest way to avoid false convergence to compare the behaviour of multiple chains that were initiated at different starting points.

Question 5:

a) The cauchyerrorpost function computes the log posterior density of a Cauchy sample with a uniform prior distribution.

b) The gibbs function implements Gibbs sampling with arbitrary log posterior density.

c) The groupeddatapost function computes the log posterior density for normal sampling where the data is observed in grouped form.

d) The indepmetrop function implements an Independent Metropolis Hastings chain.

e) The rwmetrop function implements a Random Walk Metropolis Hastings chain.